**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА АНАЛИЗА ДАННЫХ И ТЕХНОЛОГИЙ

ПРОГРАММИРОВАНИЯ

Направление: 09.03.03 – Прикладная информатика

КУРСОВАЯ РАБОТА

**По предмету «Базы данных» на тему «Оптовый магазин продуктов питания с реализацией возможности продажи в рассрочку»**

Студент 3 курса

группы 09-951

« » 2022 г. Залещин Д.С.

Научный руководитель

старший преподаватель, б/с

« » 2022 г. Еникеев И.А.

Казань – 2022

Содержание

[ВВЕДЕНИЕ 3](#_Toc104585827)

[1 . Разработка схемы БД для реализации проекта 4](#_Toc104585828)

[1.1 . Разработка логической схемы 4](#_Toc104585829)

[1.2 . Создание физической схемы 5](#_Toc104585830)

[2 . Создание скриптов для построения БД 6](#_Toc104585831)

[3 . Создание встроенной функции генерации первичного ключа. 7](#_Toc104585832)

[4. Создание триггеров для проверки документов 10](#_Toc104585833)

[4.1. Проверка ИНН. 10](#_Toc104585834)

[4.2. Проверка СНИЛС 11](#_Toc104585835)

[5. Триггер для проверки бизнес правил. 13](#_Toc104585836)

[6. Создание протокола изменений 14](#_Toc104585837)

[ЗАКЛЮЧЕНИЕ 16](#_Toc104585838)

# ВВЕДЕНИЕ

Базы данных - универсальный инструмент хранения информации для приложений, который позволяет автоматизировать работу с внесением данных, хранением, изменением и чтением. Базы данных применяются в большинстве сфер IT и являются неотъемлемой частью крупных коммерческих и пользовательских приложений. Знание основ данного инструментария является важной частью изучения программирования. Текущий курс рассчитан на изучение баз данных с целью получения навыка работы с современными СУБД.

***Цель:*** Разработка базы данных для проекта «Оптовый магазин продуктов питания с реализацией возможности продажи в рассрочку»

***Задачи:***

1. Разработать схему данных (Логическая/Физическая).
2. Создать базу данных по разработанной схеме.
3. Создать функцию генерации первичных ключей.
4. Создать триггеры для проверки корректности введённых значений СНИЛС и ИНН.
5. Разработать функцию проверки бизнес-правил.
6. Создать протокол изменений.
7. Создать функцию генерации отчётов.

# Разработка схемы БД для реализации проекта

## Разработка логической схемы

Для проектирования базы данных необходимо составить логические и физические схемы БД.

На рисунке 1 представлена логическая схема базы данных, спроектированная на платформе Erwin.

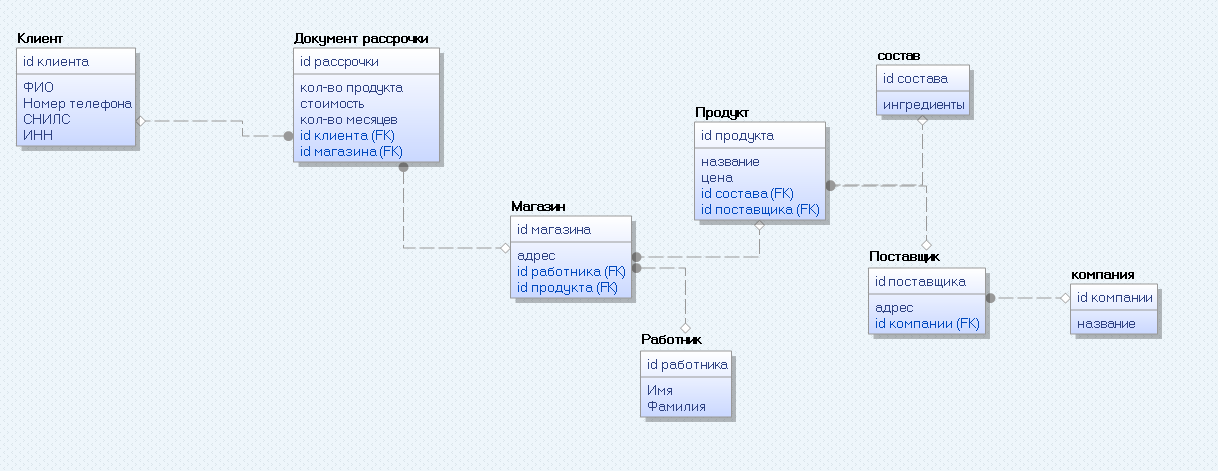


Рисунок 1. Логическая схема БД.

Схема имеет следующие сущности:

1. **Клиент** - контактная информация и информация необходимая для оформления рассрочки предоставляется в данной таблице.
2. **Документ рассрочки** - представляет из себя совокупность значений клиента и значений магазина необходимые для рассрочки. Так же имеет собственные поля, в которых описывается: кол-во взятого товара, стоимость этого товара, и кол-во месяцев, предоставленных на выплату.
3. **Магазин** - имеет значение адреса магазина и вторичные ключи на товар и работников магазина.
4. **Работник** - предоставляет информацию о работнике.
5. **Продукт** - описывает продукт, продаваемый в магазине. Имеет поля цены и названия продукта. Так же у продукта имеется состав и поставщик оформленные отдельными таблицами.
6. **Состав** - описывает множество составов одного продукта (Например, с сахаром или сахарозой и т.д.).
7. **Поставщик** - описывает адрес поставщика для какой-либо компании.
8. **Компания** - информация об производителе.

## Создание физической схемы

Отличие физической схемы от логической в том, что физическая описывает программируемую (типы данных, указание имен полей как в коде и т.д.) структуру базы данных, в то время как логическая описывает концептуальную логику проектируемой базы данных. Физическая база данных приведена на рисунке 2.

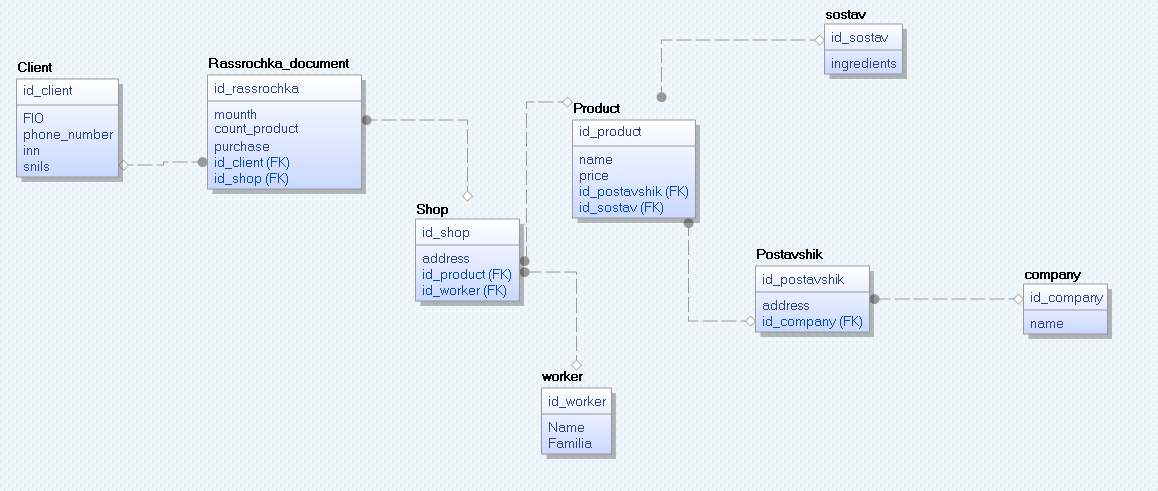
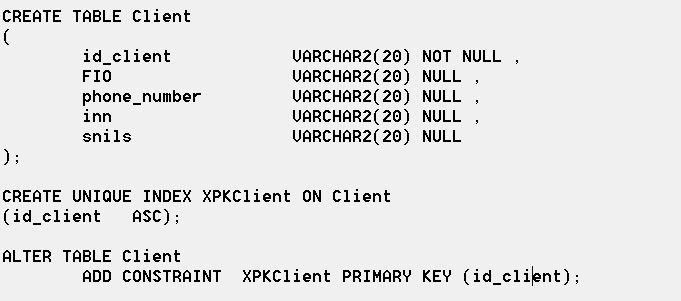


Рисунок 2. Физическая схема БД.

# Создание скриптов для построения БД

Проектируемые нами ранее схемы в приложении Erwin поддерживают такую возможность как генерация кода по созданной схеме. Спроектированная нами физическая схема уже готова к реализации БД в выбранной нами СУБД Oracle SQL. Пример сгенерированного кода приведен ниже.



Как мы видим, система сгенерировала нам таблицу по проектируемой из схемы с указанными нами типами данных и уникальными ключами.

# Создание встроенной функции генерации первичного ключа.

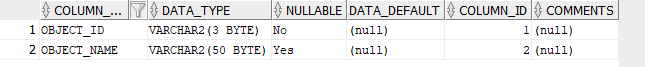
Проектируемая нами функция предоставляет функционал автоматической генерации первичного ключа для таблицы. Генерируемый функцией ключ будет иметь вид:



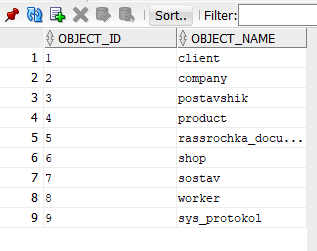
Где:

* ! – визуальный разделитель двух типов значений (может быть любым).
* Значения слева от разделителя – код таблицы (трехзначный), инициализируется нами и записывается в таблицу SYS\_OBJECT.
* Значения справа от разделителя – уникальные значении последовательности (SEQUENCE), определенной для каждой таблицы.

Таблица **SYS\_OBJECT** – таблица, содержащая в себе имя таблицы и ее id. Данная таблица будет содержать в себе имена необходимых нам таблиц, для которых мы в будущем будем генерировать ключи и имеет вид:

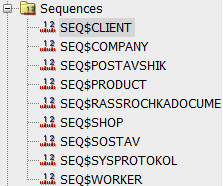


Заполнение таблицы происходит как средствами системы Oracle, так и вручную командами sql. В данном случае заполнение происходило с помощью Oracle.

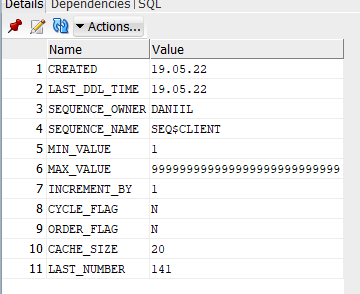


В таблицу заносятся имена необходимых таблиц и id.

Далее нам необходимо создать последовательности для каждой необходимой таблицы с значениями: начало – 0, интервал – 1, максимальное значение – 999…. (не ограничено). Указать имя последовательности необходимо начав с SEC$.... после которого указывается название таблицы заглавными буквами без пробелов:



Получая следующий пример:



После того как мы создали системную таблицу, в которой содержится id таблицы и все необходимые последовательности можно приступить к проектированию самой функции генерации. Пример части когда функции:



В качестве аргумента функция получает название таблицы, одно из тех, что мы добавили в sys\_object. Возвращает функция нам уже первичный ключ. В функции предусмотрен набор цифр и латиница как заглавная, так и прописная для формирования значения ключа.

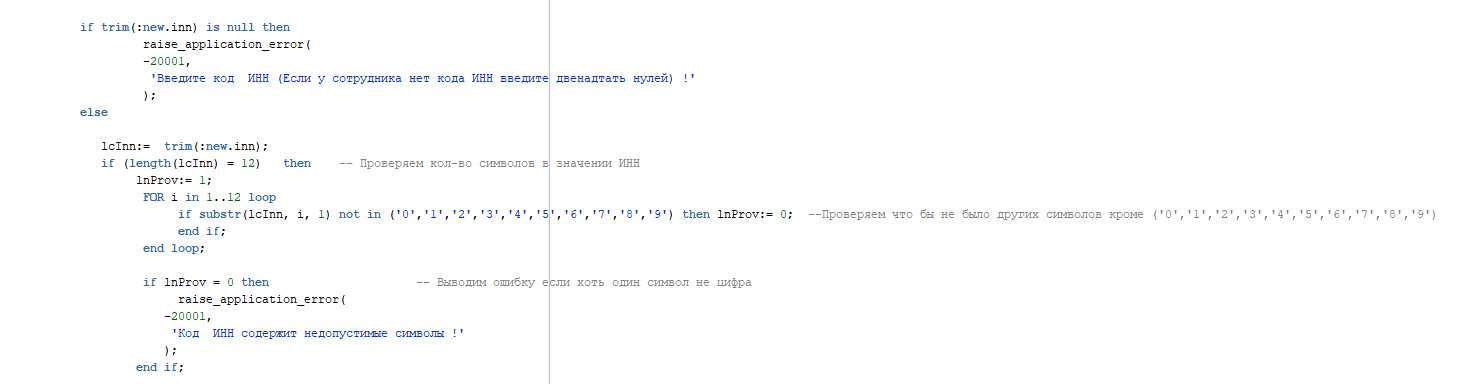
# Создание триггеров для проверки документов

# Проверка ИНН.

Для проверки вводимого ИНН используется триггер, отличие его от функции в том, что триггер вызывается сам при попытке изменения данных в таблице. Для вычисления корректности, введенного ИНН существуют правила проверки контрольных чисел:

* 1. Контрольное число n1 вычисляется как остаток от деления на 11 суммы из цифр номера (по порядку слева направо), умноженных на соответствующие (приведенные выше) коэффициенты. Если в остатке получается 10, то n1 = 0. Полученное контрольное число n1 должно совпадать с последней цифрой ИНН (как 10-значного, так и 12-значного).
  2. Контрольное число n2 - это остаток от деления на 11 суммы из цифр номера, умноженных на соответствующие коэффициенты (аналогично шагу 1). Если в остатке получается 10, то n2 = 0. Полученное контрольное число n2 должно совпадать с предпоследней цифрой 12-значного ИНН.

Фрагмент кода триггера проверки ИНН представлен ниже.





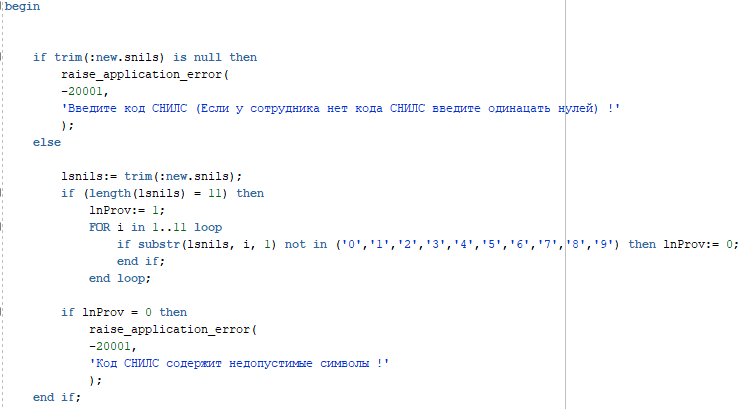
## Проверка СНИЛС

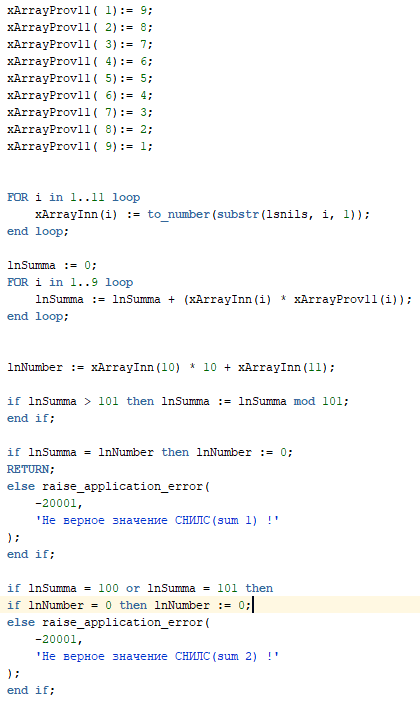
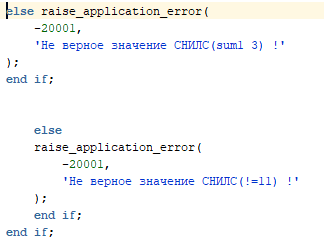
СНИЛС имеет вид "XXX-XXX-XXX YY", где XXX-XXX-XXX - номер, а YY - контрольное число.

Проверка контрольного числа имеет несколько правил:

1. Каждая цифра СНИЛС умножается на номер своей позиции (позиции отсчитываются с конца); Полученные произведения суммируются; Если сумма меньше 100, то контрольное число равно самой сумме; Если сумма равна 100 или 101, то контрольное число равно 00; Если сумма больше 101, то сумма делится нацело на 101
2. В номере XXX-XXX-XXX не может присутствовать одна и та же цифра три раза подряд.

Пример кода триггера для проверки СНИЛС представлен ниже.



# Триггер для проверки бизнес правил.

Предусмотрим, что системой будет отказано в получении рассрочки, когда клиент имеет непогашенные рассрочки суммарно на 100 000р.



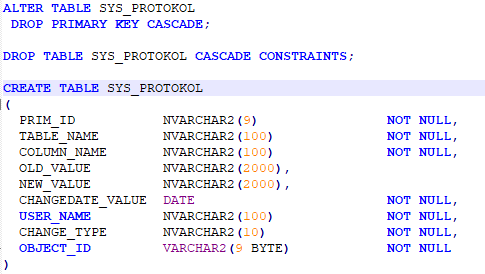
Данный триггер в начале проверяет, что данные по кол-ву продукта и цене продукта введены. Далее происходит суммирование всех рассрочек, взятых конкретным клиентом, в данном случае клиентом который собирается взять новую или еще не брал ни одной. Для клиента, который еще не брал рассрочку, предусмотрено исключение если вдруг в базе данных еще нет информации о его рассрочках.

# Создание протокола изменений

Протоколирование совершенных изменений в базе данных одна из важных частей целостности БД. Любое изменение данных необходимо фиксировать, если мы хотим знать кто, что и когда изменил. Так у нас будет полное представление об изменениях данных.

Все совершенные нами изменения будут хранится в одной таблице sys\_protokol. Данная таблица содержит в себе свой id, название таблицы, название столбца, новое значение и старое, дата изменения, имя внесшего изменения, тип выполненной операции (insert, delete update) и id первичный ключ изменяемого объекта. Для таблицы так же генерируется первичный ключ как и для остальных таблиц в базе данных.

Пример кода создания таблицы sys\_protokol:



В таблицу триггеры реагируя на изменения записывают результат.

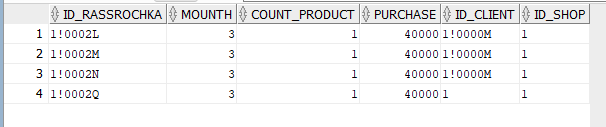
Для каждой таблицы, изменения значения которой мы хотим отслеживать, необходимо написать триггер.

Пример триггера:



Данный триггер определяет тип проводимой операции и в зависимости от типа операции протоколирует действие в соответствующую таблицу.

Пример записанной информации:



# ЗАКЛЮЧЕНИЕ

В процессе работы был изучен инструментарий для работы в СУБД Oracle, создана база данных для проекта «Оптовый магазин продуктов питания с реализацией возможности продажи в рассрочку». Использованы такие технологии, как последовательности, триггеры, функции, временные таблицы и так далее.